ABSTRACT

This paper describes the application of PROC FSEDIT in SAS/FSP and Base SAS by an intermediate size health insurance company. The need for the application arises when our data processing company's production computer is down. During these periods our data entry operators are idle, claims become backed up and substantial costs are incurred.

With the acquisition of a new computer system in our Research and Development Department we decided to set up a claims entry backup system to be used when the production computer is down. PROC FSEDIT provided the flexibility to replicate screens with over 200 entry items and edit capacity to minimize data entry errors.

The paper will examine the development and implementation of the project from screen creation to production of raw data files which are integrated back into production.

INTRODUCTION

Blue Cross and Blue Shield of Central New York is an intermediate size health insurance company which serves a ten county region in upstate New York. At the beginning of its 50th anniversary year (1986) the company employed approximately 500 people. The membership in its institutional coverage program (Blue Cross) was at 640,253 (53.9% of the market). Major medical coverage (Blue Shield) included 362,007 members (30.1% of the market). Membership in the dental coverage program was at approximately 100,000. [1]

As part of an aggressive effort to better serve its membership the company has implemented a number of cost containment and utilization review programs. A large part of the technical support for these programs is provided by the company's Research and Development Department. Much of the work produced by the Research and Development Department is done using SAS products on a mainframe computer.

BLUE SHIELD DATA ENTRY AND CLAIMS PROCESSING SYSTEMS

The Blue Shield data entry system is made up of approximately 30 data entry operators entering claims on a system which is owned and operated by private data processing company. The data entry operators key in the claims at a rate of 45 claims per hour. The claims information is transferred to the data processing company's main office located in Albany New York (150 miles) via telecommunications lines.

When the data processing company receives the claims information it processes the claims and writes the payment checks. The claims information is then turned over to Blue Cross and Blue Shield for research and analysis purposes.

Problems arise when the data processor's computer is down or when telecommunications are interrupted. During these periods the Blue Shield data entry operators are idle, claims become backed up and substantial costs are incurred. These costs are in the form of wasted time, over-time hours needed to get the system caught up, and delays in payment to subscribers and doctors.

Until the summer of 1986 the Research and Development Department's role in the claims process was limited to doing research and analysis with claims data after the claims were paid. With the acquisition of a new computer, Base SAS and SAS FULL SCREEN PRODUCT it was decided that the Research and Development Department should play a role in the beginning of the claims processing system. That role would involve the development of a Data Entry Backup System.

The goal of the project was to develop data entry screens on the
Research and Development Department's computer using PROC FSEDIT in SAS/FSP. The success of the project was dependent on how closely we could replicate the screens used in regular production. Fortunately PROC FSEDIT provided the flexibility to create the screens and also employ a number of data edits.

DEVELOPMENT OF DATA ENTRY BACKUP SCREENS

The Blue Shield claim is composed of two main parts, the "claim level" and the "service level." The claim level information appears at the top and bottom of the screen and the service level information appears in the center of the screen. The claim level information is composed of subscriber demographics, account and claim numbers, and charge information. There are also fields for information about the providers. A maximum of two providers can be entered on each claim form and for each of these providers there is a field for name or code, specialty and county.

The service level is composed of information about the procedures performed. This information includes the date range during which the particular procedure was performed, the procedure code, charge information, a provider indicator which specifies the provider (as entered at the "claim level") responsible for the procedure. There is also a special payment indicator which is used when the charges from one or more service levels are lumped into a preceding service level.

In the Blue Shield process there are four screens, since they are similar in appearance and the backup process was the same we will only use one of them for this discussion.

The first step in the development of the screens was to get an accurate listing of the data elements and the characteristics of each data element appearing on the production screens (i.e. numeric vs. alphanumeric, length and format). This step was accomplished by gathering documentation on the screens, meeting with the person responsible for training the data entry operators and getting the file layout and program used by data processing company to read the data from the screens.

After creating a list of the data elements and characteristics the next step is to create a data set containing those elements. This can be done by using the PROC FSEDIT Data Definition Screen or by building a SAS data set with no observations in it. We chose the second option for our application. Since we were working with over 200 data elements each requiring a length, informat and format specification it was easier to use a full screen editor with its copy, move, delete and change capabilities. The Data Set Definition Screen did not provide enough flexibility to handle this many data elements efficiently. For a sample of the data step see Figure 1.

An important point to keep in mind here is that the manner in which you informat and format the variables in the data step will add to the edit capabilities and appearance of the screens. For instance the SAS date informat provides an excellent edit check because FSEDIT will not allow entry of invalid dates. The informat and format are also important if you want the screens up to respond in the same way as the production screens. An example of this is found
in the field on the production screens which is numeric and zero fills to the left. This was imitated by using the zoned decimal informat and formats.

The second step in the screen development process is the modification of the screens. This is the step where we attempt to tailor the backup screens so they look exactly like the production screens. When PROC FSEDIT is run all of the variables are evenly distributed across the screen from left to right in the order in which they appear in the data set. This is the default screen (see Figure 2A-2C). To save in the screen modification phase it suggested that in the data creation step you order the variables in the order that you would like them to appear on the screen.

By using the Modification Screen in FSEDIT we were able to to customize our backup screens so that they looked exactly like the production screens (see Figure 3). For example we positioned variables in the exact locations that they appear on the production screens, key in text on the screen and had the cursor move from field to field in the same fashion as the production screens.

Another time saver we suggest is programming the function keys to perform frequently used commands. During this application the most frequently used command were text flow/text split and numbers on/off. Exercise caution when using the text flow command. This command causes every variable on the screen to flow together (with only one blank between each variable). This can cause 20 or 30 lines of variables to be lumped into 5 or 10 lines. This can be prevented by inserting blank lines where you want the flow to stop.

Once you have modified the screens PROC FSEDIT requires that you identify all of the fields that have been moved. At this point you have the option of identifying the field and making it active on the screen or not identifying and rendering it inactive. This serves the same purpose as protecting the field except that the cursor will skip over the undefined field. This may save some keying time since the operator will not have to tab or key over an unused field. Reactivating the field can be done by using the "FREE" command or by "Define variable name" on the command line, placing the cursor on the desired field and hitting enter. So the third step in the screen development process serves a dual purpose of identifying field and determining cursor movement during data entry.

The fourth step in the screen development process is the implementation of data edit checks. PROC FSEDIT's FIELD ATTRIBUTE SCREEN provides the capability of building in a number of edit checks to minimize data entry errors. For our application the most useful field attributes were the minimum/maximum and required field designations.

The minimum/maximum attribute was useful in dealing with two unusual date format problems we encountered. First, the claim number field on the production screen was composed of the julian date and a sequence number. Since SAS does not have an informat for julian dates there was no edit check for valid date. To get around this problem we divided the claim number field into three variables: julian year, julian date and sequence number. Then we placed minimum/maximum attributes on the year and date variables. This reduced the probability of data entry errors with little visible change to the screens.

The second unusual date field problem was also with dates for which SAS has no informats. On the production screens there is one date with a format of MMDDY and another with a format of MMDY. To edit the first date we split it up into three
variables with minimum/maximum attributes on each. The second date was split into two variables with the same type of attributes.

The required field attribute was also very useful as a date edit check since claims will not be paid if required fields are not filled. Since some of the information on the claim forms is not entered on the production screens the required field attribute eliminates the possibility of omitting important data.

There were a number of fields on the production screens which were specified as character variables. In several cases we knew that the possible entries for these fields were numerals so to capitalize on the edits available we used numeric variables on the backup screens, placed minimum/maximum edits on them and then converted them to character variables in the creation of the raw data files.

CREATION OF RAW DATA FILES

Once the backup screens were completed we had to devise a method of creating raw data files from the SAS data that would be collected on the backup screens. The raw data files, like the screens, had to be the same as the raw files created by the production screens. This involved more than just putting out raw data to specific locations; there were a number of challenges at this part of the process.

The first challenge was to comply with the data processor's requirement that we fill all unused character fields with binary or hexadecimal zeros and all unused numeric fields with numeric zeros. Since most of the unused fields on the record were character we decided the best way to accomplish this task was to begin each record by writing the entire record with hexadecimal zero and then write back over the record with the actual data. To satisfy the unused numeric fields we just initialized variables setting them equal zero and wrote them to the specified locations. The hexadecimal zeros are done in a loop seen in Figure 4C and again in Figure 4G. The loop in Figure 4C fills all of the the fields through the first service level. The second loop found in Figure 4F is used for any additional iteration of the service level part of the program.

The second challenge was to create variable length records to accommodate the possibility of having between one and thirteen service levels. The variable length record/multiple service level problem is handled by using arrays (Figure 4A) to process all the variables on the service levels. To write the raw records we wrote the service level part of the records by using a "DO WHILE" (Figure 4G) loop which checks to see if there is data at the current level, if there is no data the data step terminates. At the end of the service level loop there is a counter which tracks the number of the executions of the loop. The final count is then written back into the record.

The third challenge is the use of a provider indicator which points to one of the provider names that has been keyed in at the claim level. That provider information must be written out along with the service level information. There are three possible responses in the provider indicator field so there are three loops to check this indicator.

The final challenge is the special payment indicator. The special payment indicator is used when the procedure charge from one or more service levels are combined in the charge at a preceding service level. When a special indicator is encountered (value of '1') then only half of the information is written out for that service level since the information is for same the provider
CONCLUSION

Our experience with PROC FSEDIT was a good one. The only drawbacks were with changing the format of variables on the screens and the documentation. To change the format of variables we had to add a new variable to the data and then define it on the screen. The main problem with documentation is that it seems to start by describing all of the advanced techniques. We found that the chapter in the SAS/FSP should have included some of the basics such as the data creation.

As a final comment to anyone planning to use PROC FSEDIT I would strongly recommend finding out precisely what data elements (including characteristics) are going to needed for the screens. It is difficult to go back and change them.

Footnotes


* SAS, SAS/FSP are registered trademarks of the SAS Institute Inc., Cary, N.C.

For copies of this paper or information on this application please contact:

Hugh G. McCabe
Blue Cross and Blue Shield of Central New York
344 South Warren Street
Syracuse, New York 13221
(315) 424-3993
FIGURE 4B

FIGURE 4A

FIGURE 3

FIGURE 1B

FIGURE 1A

FIGURE 1C